Proof Complexity

Gabriel Diéguez Franzani

April 12, 2012

Contenidos

1. Introducción
2. Lógica proposicional y Resolución
   - Método del cuello de botella de Haken
3. Otros sistemas de demostración
   - Planos cortantes
   - Nullstellensatz y Cálculo polinomial

Introducción

- $NP =$ clase de problemas que pueden ser resueltos por una Máquina de Turing no determinista en tiempo polinomial.
- Otra caracterización: problemas para los cuales existe un *certificado* de tamaño polinomial que los pruebe.
  - Ejemplo: "esta fórmula Booleana es satisfacible".
  - Certificado: valuación
- Ahora, si consideramos $coNP$ parece que tenemos lo contrario...
  - Ejemplo: "esta fórmula Booleana **no** es satisfacible"
  - Certificado: ???
- Conjetura: $NP \neq coNP$
- Estudiaremos el tamaño de estos certificados, especialmente en problemas donde la existencia de uno pequeño no es obvia.

Algunos ejemplos

1. Inexistencia de solución para sistemas de inecuaciones lineales
Dado un sistema

$$\begin{aligned} \langle a_1, x \rangle &\leq b_1 \\ \langle a_2, x \rangle &\leq b_2 \\ \langle a_m, x \rangle &\leq b_m \end{aligned}$$

donde $a_i \in \mathbb{R}^n$ y $b_i \in \mathbb{R}$ para todo i, demostrar que no existe un vector no negativo $x \in \mathbb{R}^n$ que lo satisfaga.

Algunos ejemplos

2. Inexistencia de solución para sistemas de inecuaciones lineales enteras

Dado un sistema

$$\begin{aligned} \langle a_1, x \rangle &\leq b_1 \\ \langle a_2, x \rangle &\leq b_2 \\ \langle a_m, x \rangle &\leq b_m \end{aligned}$$

donde $a_i \in \mathbb{Z}^n$ y $b_i \in \mathbb{Z}$ para todo i, demostrar que no existe un vector no negativo $x \in \mathbb{Z}^n$ que lo satisfaga.

Algunos ejemplos

3. Inexistencia de solución para sistemas de polinomios
Dado un sistema de polinomios

$$g_1(x_1, x_2, \ldots, x_n),$$
$$g_2(x_1, x_2, \ldots, x_n),$$
$$g_m(x_1, x_2, \ldots, x_n)$$

a coeficientes reales, mostrar que el sistema $g_i(x_1, \ldots, x_n) = 0$ no tiene solución.

Algunos ejemplos

4. Contradicciones

Dada una fórmula ϕ sobre n variables proposicionales, mostrar que no existe valuación que la satisfaga.

Sobre los ejemplos

- Para los ejemplos anteriores parece no haber un certificado pequeño "obvio".
- Sin embargo, la intuición muchas veces nos engaña...

Lema (de Farkas)

Un sistema de inecuaciones lineales no tiene solución **ssi** existe una combinación de las inecuaciones que nos lleven a una contradicción;i.e., existe $y \in \mathbb{R}^m$ tal que $\sum_{i=1}^{n} y_i a_i$ es no negativo, pero $\sum_i y_i b_i < 0$.

- El tamaño de este certificado (y) es pequeño: se puede representar usando una cantidad de bits que es polinomial en la cantidad usada para representar los a_i y b_i.

Sobre los ejemplos

- Los siguientes tres ejemplos son $coNP$-hard.
- Luego, si $NP \neq coNP$, no esperamos certificados pequeños para ellos.
- No obstante, es interesante estudiar el tamaño de la demostración más corta para instancias específicas del problema.
  - Por ejemplo, **insatisfacibilidad** (o equivalentemente, certificar que una fórmula es una tautología) es un problema natural, con aplicaciones en inteligencia artificial y verificación formal, donde estamos interesados en la *tautologyhood* de una instancia cuidadosamente construida.

Sobre los ejemplos

- Es importante notar que existen lenguajes (o problemas de decisin) que **no** tienen certificados pequeños.
  - Esto está probado incondicionalmente para lenguajes fuera de $coNP$ por diagonalización.
  - Otro lenguaje famoso que no tiene certificados pequeños (de hecho, no tiene certificados **finitos**) es el lenguaje de las propiedades verdaderas sobre los números naturales en lógica de primer orden.Sí, el Teorema de Incompletitud de Gödel!)

Lógica proposicional

- Objeto de estudio: fórmula Booleana ψ.
- Nos concentraremos en el problema de mostrar que una fórmula dada ψ es una tautología.
- Por conveniencia, consideraremos el complemento de este problema: verificar que ψ, es una contradicción; i.e., mostrar que ninguna valuación σ la satisface.
- Además, restringimos nuestro estudio a las fórmulas en CNF (dado que sabemos que cualquier fórmula proposicional es equivalente a una en CNF).

Resolución

- Sea ψ una fórmula en CNF sobre variables $x_1, x_2, \ldots, x_n$.
- Sean $C_1, \ldots, C_m$ las cláusulas de ψ.
- Para $j = m+1, m+2, \ldots$, el procedimiento de resolución genera una nueva cláusula C_j implicada por las clásulas anteriores $C_1, \ldots, C_{j-1}$ usando la siguiente regla:

Regla de resolución

Dadas cláusulas C, D y una variable x_i tales que $x_i \vee C$ y $\neg x_i \vee D$ ya han sido derivadas (i.e. están en $C_1, \ldots, C_{j-1}$), entonces $C_j = C \vee D$.

- Terminamos cuando derivamos una contradicción; i.e., tanto x_i como $\neg x_i$ se han derivado, para una variable x_i.

Resolución

- Queremos probar que una fórmula ψ es una contradicción usando resolución.

Refutación por resolución

Una refutación por resolución (RPR) para ψ es una secuencia de cláusulas $C_1, \ldots, C_T$ que contiene una contradicción como la descrita anteriormente, donde $C_1, \ldots, C_m$ son cláusulas de ψ, y para $j > i$, C_j es derivada desde $C_1, \ldots, C_{j-1}$ usando la regla de resolución.

Resolución

- Cada cláusula derivada es consecuencia lógica de las anteriores, y luego el sistema es **correcto**.
  - Existe una refutación por resolución para ψ sólo si $\neg\psi$ es una tautología.
- Además, el sistema es **completo**!
  - Si $\neg\psi$ es una tautología, entonces existe una refutación por resolución para ψ.
  - De qué tamaño?

Ejercicio

Dada una CNF-fórmula ψ sobre n variables, demuestre que si ψ es insatisfacible, entonces existe una refutación por resolución para ψ de largo $2^{O(n)}$.

Resolución

- Es válido preguntarse si efectivamente existen fórmulas que requieran demostraciones tan largas.
- Puede ser que **toda** fórmula insatisfacible tenga una refutación de tamaño polinomial?
  - Dado que insatisfacibilidad es $coNP$-completo, y que creemos que $NP \neq coNP$, creemos que la respuesta es no.
  - De hecho, ahora probaremos incondicionalmente que la respuesta es **no**.

Principio del palomar

La tautología que usaremos es elemental, pero básica para las matemáticas:

Principio del palomar

- **Forma coloquial:** si tenemos m palomas y n agujeros, con $m > n$, entonces al menos un agujero debe contener más de una paloma.
- **Más formal:** no existe una función inyectiva desde un conjunto de tamaño m a uno de tamaño n, con $m > n$.

- Si bien para nosotros[1] es obvio, este principio sustenta hechos no triviales en matemáticas.
- Luego, es válido preguntarse si un sistema de demostraciones simple (como resolución) tendrá problemas demostrándolo de manera breve.

[1] Sí, hay gente para la que no es obvio

Principio del palomar

- La versión proposicional del principio consiste en una clase de tautologías $\{\neg PHP_n^m : m > n\}$, donde $\neg PHP_n^m$ es la siguiente CNF-fórmula:
  - Para $i \leq m, j \leq n$, sean variables P_{ij}, que serán verdaderas si la paloma i está asignada al agujero j.
  - Consideramos cláusulas $P_{i,1} \vee P_{i,2} \vee \ldots \vee P_{i,n}$ para cada $i \leq m$; esto es, la i-ésima paloma está en algún agujero.
  - Tenemos también cláusulas $\neg P_{i,k} \vee \neg P_{j,k}$ para cada $i, j \leq, k \leq n$, con $i \neq j$; esto es, el k-ésimo agujero no tiene al mismo tiempo a la i-ésima y j-ésima paloma.
  - Todas estas cláusulas establecen que ningún agujero contiene más de una paloma.
  - **Obs:** esta fórmula tiene $m + \binom{m}{2} n \leq m^3$ cláusulas.

Cota inferior para PHP

Teorema

Para $n \geq 2$, toda refutación por resolución de $\neg PHP^n_{n-1}$ tiene tamaño al menos $2^{\frac{n}{20}}$.

Antes de demostrar el teorema, debemos establecer algunas cosas.

- Testearemos una refutación por resolución asignando valores a las variables.
- Una refutación correcta mostrará que ninguna asignación puede satisfacer todas las cláusulas.
- Ahora, permitiremos refutaciones que muestren que un subconjunto de las posibles asignaciones no puede satisfacer todas las cláusulas.

Cota inferior para PHP

- En otras palabras, cuando sustituyamos con cualquier asignación de este subconjunto, la refutación derivará una contradicción.
- Es claro que esto es una relajación del concepto de refutación por resolución, dado que la refutación no necesariamente derivará una contradicción para otras asignaciones.
- Sin embargo, cualquier cota inferior para esta noción relajada también aplicará al caso general.

Cota inferior para PHP

- El conjunto de asignaciones que usaremos serán los mapeos que asignen $n-1$ palomas a $n-1$ agujeros uno a uno, dejando la n-ésima paloma fuera.
- Notemos que existen $n!$ tales asignaciones.
- Si el índice de la paloma que es dejada fuera es k, diremos que la asignación es k-crítica.

Cota inferior para PHP

- El conjunto de asignaciones que usaremos serán los mapeos que asignen $n-1$ palomas a $n-1$ agujeros uno a uno, dejando la n-ésima paloma fuera.
- Notemos que existen $n!$ tales asignaciones.
- Si el índice de la paloma que es dejada fuera es k, diremos que la asignación es k-crítica.

Cota inferior para PHP

- La restricción anterior simplifica la notación, dado que podemos hacer que todas las cláusulas sean *monótonas*; es decir, que no tengan variables negadas.
  - Para cada cláusula C en la demostración, construimos una cláusula *monotonizada* reemplazando cada variable negada $\neg P_{i,j}$ por $\vee_{l \neq i} P_{l,j}$.
  - Es fácil ver que la nueva cláusula es satisfacida por exactamente las mismas asignaciones que la cláusula original.
- El siguiente lema muestra que las refutaciones monotonizadas siempre deben contener una cláusula grande.

Lema

Toda refutación por resolución de $\neg PHP^n_{n-1}$, una vez monotonizada, debe contener una cláusula con al menos $2n^2/9$ variables.

Cota inferior para PHP

Lema

Toda refutación por resolución de $\neg PHP^n_{n-1}$, una vez monotonizada, debe contener una cláusula con al menos $2n^2/9$ variables.

Demostración:

- Para cada cláusula C en la refutación monotonizada, sea $witness(C) = \{i :$ hay una asignación i-crítica α que hace falsa a $C\}$
- La *complejidad de una cláusula*, $comp(C)$, es $|witness(C)|$.
- Si usamos resolución para derivar una cláusula C de dos cláusulas anteriores C', C'', entonces $comp(C) \leq comp(C') + comp(C'')$, dado que toda asignación que haga falsa a C debe hacer falsa a al menos una de C' o C''.

Cota inferior para PHP

- Luego, si C es la primera cláusula con complejidad $> n/3$, luego $n/3 < comp(C) < 2n/3$.
- Mostraremos que tal C es grande.
- Tomemos un $i \in witness(C)$ y una asignación i-crítica α que haga falsa a C.
- Para cada $j \notin witness(C)$, consideremos la asignación j-crítica α' obtenida al reemplazar i por j.
  - Esto es, si α mapeaba la paloma j al agujero l, entonces α' deja a j fuera y mapea la paloma i al agujero l.
- Dado que $j \notin witness(C)$, esta asignación j-crítica debe satisfacer C, y luego concluimos que C contiene la variable $P_{i,l}$.

Cota inferior para PHP

- Digamos que $comp(C) = t$.
- Hacemos lo mismo sobre todos los demás $n - t$ valores de $j \notin witness(C)$ usando el mismo α, con lo que obtenemos que C contiene $n - t$ distintas variables del tipo $P_{i,l}$.
- Repitiendo el argumento anterior para cada $i \in witness(C)$, concluimos que C contiene al menos $t(n - t)$ variables.
- Luego, como $n/3 < t < 2n/3$, tenemos que $t(n - t) > 2n^2/9$, lo que prueba el lema. $\square$

Cota inferior para PHP

Ahora podemos probar el teorema:

Teorema

Para $n \geq 2$, toda refutación por resolución de $\neg PHP^n_{n-1}$ tiene tamaño al menos $2^{\frac{n}{20}}$.

Demostración:

- Digamos que una cláusula en la refutación monotonizada es grande si tiene al menos $n^2/10$ variables.
- Sea L es número de cláusulas grandes. Por el lema anterior, $L \geq 1$.
- Definimos una restricción sobre algunas variables que reduce drásticamente el número de cláusulas grandes.
  - Por *averaging* sabemos que hay una variable $P_{i,j}$ que aparece en $1/10$ de las cláusulas grandes.
  - Restringimos las variables con $P_{i,j} = 1, P_{i,j'} = 0$ para $j' \neq j$ y $P_{i',j} = 0$ para $i' \neq i$.

Cota inferior para PHP

- Lo anterior hace que todas las cláusulas monotonizadas que contienen a $P_{i,j}$ se hagan verdades, por lo que pueden removerse de la demostración por resolución.
- Esto deja a lo más $9/10L$ cláusulas grandes.
- En otras palabras, por contención una paloma y un agujero han sido removidos por la restricción, con lo que ahora tenemos una demostración por resolución monotonizada para $\neg PHP^{n-1}_{n-2}$.
- Repitiendo este paso $t = \log_{10/9} L$ veces, obtenemos una demostración por resolución monotonizada para $\neg PHP^{n-t}_{n-1-t}$ que **no** contiene cláusulas grandes.
- Entonces, si $L < 2^{n/20}$, se tiene que $t < n/3$, y por lo tanto tenemos una refutación monotonizada para $\neg PHP^{n-t}_{n-t-1}$ sin cláusulas más grandes que $n^2/10$.
- Pero esto es menor que $2(n-t)^2/9$, lo que contradice el lema. $\square$

Otros sistemas de demostración: Planos cortantes

- Este sistema ataca el problema de certificar no-factibilidad de un sistema de inecuaciones lineales con coeficientes y variables enteras.
- Ya vimos que este problema es $coNP$-completo.
- Por ejemplo, dada cualquier fórmula ϕ en 3CNF, podemos representarla con un sistema tal que la fórmula es una contradicción ssi el sistema no es factible.
  - Para cada variable x_i en ϕ tenemos una variable entera X_i que satisface $0 \leq X_i \leq 1$ (i.e., $X_i \in \{0, 1\}$).
  - Para una cláusula $x_i \vee x_j \vee x_k$ tenemos la inecuación lineal $X_i + X_j + X_k \geq 1$.
  - Si aparece una variable negada, usamos $1 - X_i$.

Planos cortantes

- Dado un sistema de inecuaciones lineales enteras no factible, el sistema de Planos cortantes prueba la no factibilidad derivando la desigualdad $0 \geq 1$ en un número finito de pasos.
- Produce una secuencia de desigualdades $l_1 \geq 0, l_2 \geq 0, \ldots, l_T \geq 0$, donde la r-ésima desigualdad es:
  1. Una inecuación que aparece en el sistema,
  2. $\alpha l_u + \beta l_v \geq 0$, donde α, β son enteros no negativos y $u, v < r$ o
  3. Una derivación desde algún l_u con $u < r$ usando la siguiente regla: si l_u es de la forma

     $$\sum_{i=1}^{n} a_i x_i - b \geq 0$$

     donde los números $a_1, \ldots, a_n$ tienen un máximo común divisor $D \geq 2$, entonces la nueva inecuación es

     $$\sum_{i=1}^{n} \frac{a_i}{D} x_i - \lceil \frac{b}{D} \rceil \geq 0$$

Planos cortantes

- Qué pasa cuando D no es divisor de b?
- Se sabe que existe un Teorema de interpolación factible para Planos cortantes.
- Este teorema se ha usado para probar cotas inferiores exponenciales para el sistema.

Nullstellensatz y Cálculo polinomial

- Ahora atacamos la no factibilidad de sistemas de ecuaciones definidos por polinomios.
- Igual que antes, usaremos 3CNF:
  - Para cada variable x_i en la fórmula, tendremos una variable X_i y una ecuación $X_i^2 - X_i = 0$, asegurando que $X_i \in \{0, 1\}$.
  - Podemos transformar cada cláusula en una ecuación de grado 3. Por ejemplo, la cláusula $x_i \vee x_j \vee \bar{x}_k$ se transforma en la ecuación $(1 - X_i)(1 - X_j)X_k = 0$.

Hilbert's Nullstellensatz

Un sistema de ecuaciones $p_1(X_1, \ldots, X_n) = 0, \ldots, p_m(X_1, \ldots, X_n) = 0$ en un campo $\mathbb{F}$ no es factible ssi existen polinomios $g_1, g_2, \ldots, g_m$ tales que

$$\sum_i g_i(X_1, \ldots, X_n) p_i(X_1, \ldots, X_n) = 1$$

Nullstellensatz y Cálculo polinomial

- El sistema de demostración Nullstellensatz considera:
  - Los p_i como axiomas.
  - Una secuencia de g_i que satisfacen la ecuación anterior como prueba de no factibilidad.
- El teorema de Hilbert prueba que este sistema es correcto y completo.
- El tamaño de la demostración es el número de bits requerido para codificar los coeficientes de los polinomios.

Nullstellensatz y Cálculo polinomial

- El cálculo polinomial es similar, pero los g_i pueden ser calculados por un programa en lugar de especificar todos los coeficientes.
- Concretamente, una refutación en cálculo polinomial es una secuencia finita de polinomios $f_1, \ldots, f_T$ tales que cada f_r es:
  1. Un p_i,
  2. $\alpha f_u + \beta f_v$, con α, β constantes y $u, v < r$, o
  3. $x_i f_u$ donde x_i es una variable y $u < r$.
- El tamaño de la refutación es T y el grado es el máximo grado de algún f_u.
- Se han probado cotas inferiores exponenciales para ambos sistemas.